An API, or Application Programming Interface, is essentially a set of rules and protocols that allows different software applications to communicate with each other. It defines the methods and data formats that developers can use to interact with a particular software component or service.

In the context of Google Maps, the Google Maps API allows developers to integrate Google Maps functionality into their own applications or websites. This means developers can access features like displaying maps, adding markers, getting directions, and more, all within their own applications.

Here's a basic overview of how the Google Maps API works:

**Registration and Authentication**: Before using the Google Maps API, developers typically need to register for an API key. This key is a unique identifier that authenticates requests to the Google Maps servers.

**Integration**: Once developers have obtained an API key, they can integrate Google Maps into their application using the provided JavaScript library or other supported programming languages.

**Making Requests**: Developers can use the various methods provided by the API to make requests for map data, such as displaying a map on a webpage, adding markers to indicate specific locations, or getting directions between two points.

**Receiving Responses**: When a request is made to the Google Maps API, the server processes the request and sends back a response containing the requested map data. This data is typically in a format such as JSON (JavaScript Object Notation) or XML (extensible Markup Language).

**Rendering**: Finally, the application renders the received map data to display it to the user. This could involve drawing the map on a webpage, placing markers at specified coordinates, or displaying route information.

Overall, the Google Maps API provides developers with a powerful toolset for incorporating mapping functionality into their applications, allowing them to create customized and interactive map experiences for their users.

A UML (Unified Modeling Language) diagram is a standardized visual representation used in software engineering to illustrate the structure and behavior of a system or software application. UML diagrams help developers and stakeholders to understand, design, and communicate various aspects of the system in a clear and concise manner.

There are several types of UML diagrams, each serving a specific purpose:

**Class Diagram**: Shows the static structure of a system by depicting the classes, attributes, methods, and relationships between them.

**Use Case Diagram**: Illustrates the interactions between users (actors) and the system, focusing on the system's behavior from an external perspective.

**Sequence Diagram**: Represents the interactions between objects over time, typically used to visualize the flow of messages and method calls in a system.

These diagrams serve as blueprints for developers, enabling them to plan, design, and implement software systems effectively. UML diagrams are also valuable for stakeholders, such as clients and project managers, as they provide a visual representation of the system's architecture and functionality, facilitating communication and decision-making throughout the development process.

GUI’s.

1. Signup page:

Name

Phone number (OTP)

Email

Signup button

Forgotten Password

Login button

1. Login page:

Name

Phone number

Login button (Verification)

1. Home Page:

Map view

Direction

Profile

Search bar

1. Direction:

Map view

Share your Position

1. Profile:

Edit Information

1. Search bar.

CLASS DIAGRAM:

* GWAGSI RAWLINGS
* ABINGSEH CINDY

USE CASE DIAGRAM:

* CHII FAVOUR
* DJIMPE JULY

SEQUENCE DIAGRAM:

* KIMIA BATHSON
* EKO PEARLEEN

OBJECT DIAGRAM:

* KOUAM LOIC